Substitute Specification (marked-up version showing changes)

Data Processing Circuit, Multiplier Unit With Pipeline, ALU and Shift Register Unit For Use In A Data Processing Circuit

BACKGROUND OF THE INVENTION

[0001] Nowadays a large number of personal computers make use of processors which have a complex instruction set (CISC). Such processors are provided with a central processing unit, the function of which is adjusted at each clock pulse to perform the desired operation on two operand words. These processors are currently commercially available under Intel code numbers beginning with 80.

[0002] Although the clock speed of the function adjustable processors has been increased considerably, the organizational structure of such a processor forms a great obstacle to further increase the processing speed. For instance, during multiplying and dividing of two operand words, frequent use must be made of registers internally present in the computer.

[0003] So-called work stations often use a pipeline structure with a reduced instruction set (the so-called RISC, Reduced Instruction Set Computer) in order to increase the speed of the work station. This structure provides an increase in speed of so-called vector operations, wherein a large number of data words have to be subjected to the same arithmetic operation. Since a limited instruction set can be implemented efficiently, the execution of a large number of instructions requires only a single clock pulse.

[0004] Although the RISC structure achieves an increase in speed for frequently occurring operations (such as multiplications) more complex instructions for particular operations are omitted from the instruction set and ~~and~~, therefore, the speed of

executing such operations is not increased. In addition, the processing unit is often designed for data words with a fixed word length, for example 32 or 64 bits.

[0005] In EP-A-0173383, a processor for floating point operations is disclosed. Such floating point operations are not useful for image or graphical processing applications, where operations have to be performed on integer data words of 8, 16, or 32 bits.

[0006] In the article "The 1860TH 64-bit supercomputing microprocessor" by L. Kohn et al, published in the proceedings of supercomputing, 13-17 November 1989, Reno, Nevada, VS, 1989, IEEE Computer Society Press, Washington D.C., a RISC based micro-processor for executing multiplications for either 64 bit or 32 bit words is described. As described above, such RISC concept does not provide for increased speed when integer data words of 8 bits or multiplies thereof have to be processed.

[0007] Also, in EP-A-0380100, a multiplier is disclosed for processing 32 bit operands to provide two 16 bit by 16 bit fixed point products for one 32 bit floating point product during each clock cycle.

[0008] For image and/or graphics processing applications however, operations have to be performed on data words of 8 or 16 bits or a number of mutually associated bytes before even a limited speed increase is achieved in the RISC concept.

[0009] The present invention provides a data processing circuit comprising:

- a multiplier unit for multiplying integer data words of 8 bits or multiples thereof having a pipeline and in which the word length is adjustable for multiplying the integer data words;
- an arithmetic logic unit (ALU) having an adjustable word length for performing arithmetic operations on integer data words of 8 bits or multiples thereof;

- a register unit provided with at least two registers for storing the integer data words of 8 bits or multiples of 8 bits on which the operation and/or pipeline multiplication has to be performed; and

- a bus structure which comprises a number of separate buses and which effects the transport of integer data words from and to the multiplier unit, the arithmetic logic unit and the register unit.

[0010] The data processing unit according to the present invention achieves a speed, for graphic applications, that is more than twice as great as in existing systems. In contrast to RISC and CISC, the data flow between the above specified circuits (multiplier, ALU etc.) is not fixed. Rather, the programmer is free to program the sequence of the data flow through the different units (free pipeline).

[0011] The present invention further provides a multiplier unit with a pipeline for use in a data processing circuit.

[0012] The present invention also comprises an arithmetic logic unit for use in a data processing circuit.

[0013] Finally, the present invention provides a shift register unit for use in a data processing circuit.

BRIEF DESCRIPTION OF THE DRAWINGS

[0014] Further advantages, features and details of the present invention will be elucidated on the basis of the following description of a preferred embodiment thereof with reference to the annexed drawings, in which:

[0015] Fig. 1 shows a functional diagram of a graphic application of a data processing circuit according to the present invention;

[0016] Fig. 2 shows an outline diagram of the data processing circuit of Fig. 1;

[0017] Fig. 3 shows a functional diagram of the internal structure of the data processing circuit of Fig. 1;

[0018] Fig. 4 shows a first functional diagram of the arithmetic logic unit of the diagram of Fig. 3;

[0019] Fig. 5 shows a second functional diagram of the arithmetic logic unit of the diagram of Fig. 3;

[0020] Fig. 6 shows a functional diagram of the multiplier unit with pipeline of the diagram of Fig. 3;

[0021] Fig. 7 shows a functional diagram of a Wallace tree in the diagram of Fig. 6; and

[0022] Fig. 8 shows a functional diagram of the shift register unit from the functional diagram of Fig. 3.

DETAILED DESCRIPTION OF THE INVENTION

[0023] A data processing circuit 1 (Fig. 1) according to the present invention, also named DISC or IMAGINE, is coupled via a bus 2 to a data memory 3, for instance SRAM (Static Random Access Memory). The data processing circuit 1 is further connected via a bus 4 to a main or video memory 5 for storage of image data, which is constructed from DRAM (Dynamic Random Access Memory) cells or is a (more expensive) VRAM. This main memory 5 drives a RAMDAC (Random Access Memory for a Digital Analog Converter) 7 via bus 6, which in turn provides a monitor (not shown) with the color signals R (red), G (green) and B blue).

[0024] In practical applications the data processing circuit 1 will be coupled via a buffer 9 and access logic 10 to a host processor (not shown). The configuration of Fig. 1 is preferably further provided with an instruction RAM 11 which is coupled via a bus 12 to the data processing circuit 1 as well as via a buffer 112 in which registers and drive means are incorporated. A clock means 13 provides the diverse components of the configuration with clock signals while a circuit 14 is included in the con- figuration for the video timing. A video input circuit 15 is preferably connected to the bus 6 for feeding video signals to the image memory 5.

[0025] The structure of the data processing circuit is shown schematically in Fig. 2 and comprises a parallel multiplier 20 which comprises a RAM 21, an accumulator 22 and a multiplier and Wallace tree 23. The data processing circuit also comprises a data input and output circuit 24, a parallel shift register 25, a bus structure 26, a circuit 28 for unary operations, a circuit 29 for driving the image memory, a circuit 30 for image input and output, an arithmetic logic unit 31, a circuit 32 for driving the register bank and a vector index generator, a register bank 33, a mask generator 34 which comprises a

transparent mask 35, an opaque mask 36, a window mask 37, a line mask 38, a polygon mask 39, a mask assembly means 40 and a range check 41, a circuit with phase-locked loop 42 and a circuit 43 for instruction processing which comprises a program control 44, start-up ROM 45 and an interrupt processing means 46.

[0026] The bus structure 26 (Fig. 3) comprises a control SC-bus 51, an A-bus 52, a B-bus 53, a Q-bus 54, an F-bus 55, an M-bus 56, a U-bus 57, a D-bus 58 and a V-bus 59, each of which is, for instance, 32 bits ide. Each of several functional units of data processing circuit 1 drives its own output bus and has a separate, dedicated output (bus) register/driver for its bus, which can be read in the following cycle by various other functional units. For example, multiplier 23 drives the M-bus 56 using its bus register, M-reg 66; ALU 31 drives the F-bus 55 using its bus register, F-reg 64; shift register 25 drives the Q-bus 54 using its bus register, Q-reg 62; register bank 33 drives the A-bus 52 and B-bus using bus registers, A-reg 60 and B-reg 61, respectively; image input and output circuitry 30 drives the V-bus using its bus register, V-reg; and so forth. This approach allows parallel processing for all of the functional units.

[0027] The register bank 33 is connected via output registers 60 and 61 to the A-bus and B-bus respectively. Register bank 33 contains ninety-six inputs which are single 32 bit, double 16 bit or quadruple 8 bit words. Three ports enable simultaneous performance of two read actions and a write action. Sixty-two of the ninety-six registers are directly accessible. The remaining thirty-two inputs are addressed via the vector index generator 32 which can generate a maximum of 12 locations per cycle (i.e., four byte sections for each of the three ports, since each word segment can be selected separately within the registers).

[0028] The parallel shift register 25 is designed such that it can shift 32 bits of data anywhere from 1 to 32 positions to the left or right in one clock cycle based on the information received via the A-bus 52. The information can be grouped into one, two or four sections of 32, 16, and 8 bits respectively. The shift can take place logically (unsigned), numerically (signed) and rotatingly. The operands are received from the B-bus 54 or the F-bus 55. The parallel shift register 25 is connected via a register 62 to the Q-bus 54. Fig. 8 schematically shows an example of a two step rotation of a 32 bit word (consisting of two 16-bit bytes) through 11 bits in a positive direction by way of four 8 bit rotations and eight 4 bit crossings.

[0029] With reference to Fig. 3, the arithmetic logic unit 31 (ALU) is connected to the A-bus 52, the Q-bus 54, the M-bus 56, the D-bus 58, the U-bus 57, the B-bus 53, the F-bus 55, again to the U-bus 57 and the V-bus 59. All the usual logic operations of a conventional ALU can be performed by the ALU 31 of the present invention in addition to numerical functions such as addition, subtraction, increment and decrement. The ALU 31 is further provided with a so-called parametric logic function. On the basis of the content of an 8 bit register, the ALU 31 can perform a random combination of 256 possible logic operations on 3 operands. The standards for X-window and MS-windows specify that logic and graphic operations must be possible in any combination. The parametric function can also be used to realize shifting, masking, combining or comparing operations in a single clock cycle.

[0030] The ALU 31 can be adjusted as a single, double or quadruple parallel unit for 32, 16 and 8 bit operands respectively. The data coming from the A-, Q-, -- or D-buses determines the selection of the size of the operands to be processed. A mode selector 63 is connected to the ALU 31 and generates a status signal on output 64. The ALU 31

is further connected to the F-bus 55 via an output register 64. Fig. 4 shows a functional diagram of the ALU for a parallel quadruple operation on operands of 24 bits, while Fig. 5 shows a functional diagram of a double operation with 48 bit operands. In Fig. 5, two selectors and two accumulators, each of 8 bits, are combined.

[0031] The multiplier 23 is embodied as pipeline with five clock cycles. The multiplier is capable of performing pipeline operations on 32 bit, 16 bit and 8 bit words. All possible multiplication operations with numbers, signed and unsigned, or a combination thereof, in addition to execution of the multiplication of 16 bit complex numbers and 8 bit matrices with vectors is possible due, *inter alia*, to the presence of a Wallace tree (Fig. 7). The multiplier operates internally with 48 bit results or double 24 bit or quadruple 12 bit values, two of which are transported simultaneously via 96 bit data channels. Fig. 6 shows a functional diagram of the multiplier with five clock levels. The multiplier is connected to the M-bus 56 via an output register 66.

[0032] The circuit for unary operations 28 converts data, for instance, binary to unary (linear), indicates the position of the most significant bit, determines the absolute value of a sign and reverse the bit sequence of a word. Circuit 28 can operate on a word of 32, 16 or 8 bits.

[0033] The mask generator 24 has a number of independent sub-units. The window mask 37 determines which regions the other operations must fall. The circuit 41 for range checking operates on the basis of pre-defined patterns and, therefore one of its most important applications is generating letter characters. The circuit 41 also serves to check three-dimensional pixel data, such as depth and color.

[0034] The line mask 38 generates a horizontally defined pattern between a predetermined beginning and end. The line mask 38 can generate up to four lines

simultaneously and supports, for instance, the creation of polygons. A shape along a horizontal line of the image can be produced using the line mask 38, when no interruptions occur along the line.

[0035] The polygon mask 39 serves to generate elements for which the line generator is not suitable, for instance, Chinese characters. The polygon mask 39 defines the number of contour transitions on the horizontal lines passing through a relevant pixel.

[0036] The mask assembly 40 performing the function of overlaying diverse masks. The results from the mask assembly 40 is transmitted to the respective transparent and/or opaque masks 35, 36 where the actual image for display is created. The transparent and opaque masks 35, 36 can both contain a maximum of 128 pixels in a matrix of 4 x 32.

[0037] The circuit for data input and output 24 is connected to a 32 bit data channel and a 32 bit address bus. The range for addressing comprises 32 Mbyte.

[0038] The entry of instructions takes place under the control of the program control unit 44. With a 22 bit address, a following instruction word is continuously assigned which is subsequently entered via a separate 64 bit bus. The program memory can have a size of 4M x 64 bits.

[0039] The drive of the image memory 29 is adapted to generate an address on the basis of an X/Y position so that any random image segment can be addressed on the basis of its location and the image in the image memory. The image memory is also suitable for storing other data banks such as lists and data banks with graphic elements.

[0040] When a clock frequency of 66 MHZ is used for a data processing circuit according to the present invention, it is possible to operate system such that the access time for the memory is 70 ns.

[0041] The data processing circuit 1 can be programmed in a higher program language, such as C, so that it is easily programmed, as in RISC and CISC processing units. The data processing circuit 1 can be programmed with instructions according to the RISC concept as well as with the CISC instructions of a personal computer. In order to achieve a large increase in speed for graphic applications, the programmer can program all functions of the data processing circuit 1 at a lower level via an instruction field of 64-bits. The ALU 31 and the multiplier unit can be set to parallel operations, whereby the speed for graphic applications can be increased by a factor of 4-20 as compared to existing RISC processors. For a particular application, a programmer will set a "once-only" series of instructions and control registers. Subsequently, the programmer will start the processor with one command, hereafter the processor independently processes the pixel flows.

[0042] As example of the speed increase which can be gained by way of the present invention, algorithm consisting of five instructions for rotating and interpolating a color image is presented which can accommodate a total of 38 instructions, that is:

read 2 x 16 bit register;

increment 2 x 16 bit register address;

read 1 x 10 bit constant;

shift 2 x 16 bit word;

read 2 x 16 bit constant;

add 2 x 16 bit value;

read out 4 x 8 bit 2D memory data;

read out 4 x 8 bit image memory data;

increment 1 x 32 bit image memory address;

multiply 4 x 8 bit value;

read 4 x 12 bit accumulator register;

accumulate 4 x 12 bit value write 4 x 12 bit accumulator register; and

increment 2 x 5 bit register address accumulator.

[0043] The data processing circuit according to the present invention can be built into specific equipment but can also be embodied as an extension card for a personal computer. Owing to the flexible utilization of the hard- ware, even at lower clock speeds than, for instance, 200 MHZ, which is currently among the highest, from 5 to 20 times improvement in image processing speed can be obtained. This makes the data processing circuit according to the present invention suitable for real-time video operations and so-called virtual reality.

[0044] The data processing circuit 1 has an hierarchical instruction set to provide compatibility with existing software, compilers and operating systems at one end, and highly efficient parallel vector processing at the other end. The various instruction levels can be intermixed freely in the assembler code. All instruction levels can be written as "in line" code in C and C++ programs, for example.

[0045] The hierarchical instruction set has the following instruction levels:

- Compiler based C code and C++ code (compiler generated)
- RISC/CISC level assembly code (macro function set)
- Free pipeline assembly code (native machine language: graphs of pipeline sequences)

- Vector processing level (set of macro functions for vector operations and user defined vector operations)
- Specific use of control registers and special purpose graphics hardware.

Free Pipeline Assembly Code Level

[0046] As an example of the free pipeline assembly code level, the ALU instructions include those listed in the following table:

Table 1. ARITHMETIC AND LOGIC UNIT INSTRUCTIONS
(All functions executed at a single cycle throughput)
(Modes: all functions operate on 4x8 bit, 2x16 bit and 32 bit)

Mnemonic	Operation	Mnemonic	Operation
F = clear	F = all bits '0'	**F = decr (R)**	F = R - 1
F = and (R,S)	F = R and S	**F = incr (R)**	F = R + 1
F = andrev (R,S)	F = (not R) and S	**F = decr (S)**	F = S - 1
F = copy (S)	F = S	**F = incr (S)**	F = S + 1
F = andinv (R,S)	F = R and (not S)	**F = subdecr (R,S)**	F = R - S - 1
F = noop (R)	F = R	**F = sub (R,S)**	F = R - S
F = xor (R,S)	F = R xor S	**F = subbor (R,S)**	F = R - S + carry
F = or (R,S)	F = R or S	**F = minus (S)**	F = - S
F = nor (R,S)	F = not (R or S)	**F = subdecr (R,S)**	F = S - R - 1
F = equiv (R,S)	F = R xnor S	**F = sub (R,S)**	F = S - R
F = invert (R)	F = not R	**F = subbor (R,S)**	F = S - R + carry

F = orrev (R,S)	F = (not R) or S	**F = minus (R)**	F = - R
F = copyinv (S)	F = not S	**F = add (R,S)**	F = R + S
F = orinv (R,S)	F = R or (not S)	**F = addincr (R,S)**	F = R + S + 1
F = nand (R,S)	F = not (R and S)	**F = addcar (R,S)**	F = R + S + carry
F = set	F = all bits '1'	**F = logic (R,S)**	F = three op logic function

Operand **R** can be selected from busses: **A**(register), **D**(data-memory), **M**(multiplier) and **Q**(barrel-shifter)

Operand **S** can be selected from busses: **B**(register), **V**(image-memory), **F**(ALU) and **U**(unary function unit)

Vector Processing Level

[0047] Regarding the vector processing level, processing vectors or streams of data means that an instruction is repeated a number of times. Typically this will range from 8 to 32 times in continuous bursts, up to several million times in repeated bursts. In this situation there is no need for the instruction to be supplied on each and every cycle.

[0048] The data processing circuit 1 can have hundreds of bits devoted to extended instructions which are stored in control registers located within the various functional units. The basic 64 bit instruction word can select extended functions which use information stored in these control registers. The actual instruction word length for these extended operations is thus much longer.

[0049] In contrast with the ineffective functional units found in standard RISC and CISC processors, this instruction level makes the data processing circuit 1 an ultra high-

speed heterogeneous multi-vector processor that can perform intelligent conditional operations on parallel streams of data.

[0050] Vector processing can be implemented with the following properties of the data processing circuit 1: the Repeat Instruction, vector processing functional units, and vector type data storage access.

The Repeat instruction

[0051] During the processing of a variable length vector, the same operation is repeated a variable number of times. The repeat function fulfils this task. For example, it can target an instruction 3 to 19 cycles later, which is then repeated from 1 to 32 times. Larger vectors can be subdivided into smaller sized ones.

Vector processing functional units

[0052] All data processing units such as the ALU, the Barrel shifter and the Multiplier/Accumulator can perform vector operations. Each one can perform operations each cycle and the units are interconnected by the above-described flexible bus structure which allows a pipeline to be set up from the reading of the operands, through various functional units to the writing of the results.

Vector type data storage access

[0053] All types of data storage known to the data processing circuit 1 have a vector access mode. There are four types of data storage: the Image memory, the Data memory, the three-port register file, and the Multiplier/Accumulator register file.

The Image Memory.

[0054] Vector accesses to Image memory are performed with the **image vector** which can access vectors from 1 to 32 words. Larger vectors can be subdivided into smaller ones in co-operation with the repeat function.

The Data Memory Interface.

[0055] The Vector access for this unit is the **again** function which reads or writes from the same, the incremented or the decremented address over a range of 256 addresses. The **again** function should be part of the vector operation which is executed multiple times by the repeat operation.

Three-Port Register File.

[0056] Reading from both read ports and writing can be done with the Vector address generator. This unit has many addressing modes from simple post-increment access to offset and conditional offset/increment access modes. Typical read and write operations will use the register indices generated by the vector index generator. Some examples include the following:

A=rd4x8(ri), wr(ri,Q);

AB=rd2x16(ri,ri), wr4x8_0111(ri,D);

New indices are calculated with a generate instruction:

gen4x8ad(); gen2x16ad();

[0057] An indexed read and/or write functions and a generate function should be part of the vector operation which is executed multiple times by the repeat operation.

The Multiplier/Accumulator register file

[0058] This register file belonging to the Accumulator can be red, written and post-incremented during accumulations for linear or higher order difference operations or to store the result from a vector which comes from the Multiplier. A stored vector can be added to a newly processed vector. The control of these operations is the responsibility of the multiplier control registers. The multiple repeated vector instruction should issue an instruction during this cycle which implies the use of these control registers. Examples include the following:

read_ram; **write_ram;**
inproduct (Mb); **matrixvect (Mb);**
linearstep; **macb(Ma,Mb);**
mcon(Mb);

Control Registers

[0059] As noted above, another instruction level of the data processing circuit 1 is the specific use of control registers and special purpose graphics hardware, which can be used to extend the basic instruction set. Using bits in the control registers, individual instruction words can be extended on a per-instruction-cycle basis. This level will be described now by example, with reference to the multiplier 20.

[0060] The extended function set includes the more specialized functions such as the 16-bit complex multiplication, the 4x4 matrix times vector multiplication, etc. Many of them can also use the accumulator stage and the range clip stage of the MAC 20.

[0061] The two multiply/accumulate instructions, M=macs() and M=macb(), use multiplications from the basic set but allow the use of the accumulator stage and the range clip stage.

[0062] Table 2 below shows an example of the multiplier functions of the extended set. Table 3 indicates which resources a given instruction can use. It defines which fields of the MAC Control register cr12 are used during the execution of the instruction.

[0063] Some fields refer to two other registers containing control information. The MAC Rampointer register (cr13) is used when the **read** or **write** fields indicate an Access to the Accumulator Ram file. This register contains read and write pointers plus some other fields concerning the Ram file. The Range Unit control register (cr16) is used if the Range Unit field (**range**) is TRUE (logical '1'). The Range Unit control register uses two registers with a Low and High limit (cr17 and cr18).

[0064] The data **size**, **type** and **sign** fields define the data format used for a certain multiplication function, e.g.: the matrix x vector product can be performed in 16 different ways.

[0065] The **accu** field selects between the three operands for the Accumulator: the Multiplier result, the Accumulator Ram file and the Accumulator itself.

[0066] The **pipe** field controls the two 4x4 register sets for the matrix operations. These are located in the first stage where the multiplier input operands are selected.

Table 2

46...42		**Mnemonic**	**Operation (single cycle throughput)**
10000	10	M=inproduct(Mb)	quadruple vector inproduct (4 x 8 bit vectors)
10001	11	M=matrixvec(Mb)	4x4 matrix vector multiplication (8 bit)
10100	14	M=loadpipe(Ma,Mb)	shift 4x4 matrix data and coefficient pipelines
10101	15	M=read_ram()	read 96 bit word from the accu Ram into accumulator
10110	16	M=write_ram()	write 96 bit word from the accumulator to accu Ram

10111	17	M=linearstep(*)	incremental add (Gouraud, Zbuffer..) 4x24bit, 2x48bit
11000	18	M=macs(Ma,Mb)	multiply accumulate (scalar)
11001	19	M=macb(Ma,Mb)	multiply accumulate (block)
11100	1C	M=vectprod(Ma,Mb)	16 bit vector dot product and cross product
11101	1D	M=complex(Ma,Mb)	16 bit complex multiply M = a*b - b*c + i(a*d + b*c)
11110	1E	M=nop	no operation
11111	1F	M=halt	halt MAC: freeze the entire MAC pipeline.

Table 3

			Fields of MAC Control Register (cr12)								
			write	read	size	type	sign	accu	out	range	pipe
46...42		**Mnemonic**	30-29	27-26	21-22	19-18	17-16	15-12	10-8	7	4-0
10000	10	M=inproduct(Mb)	used	used	(4x8)	used	used	used	used	used	used
10001	11	M=matrixvect(Mb)	used	used	(4x8)	used	used	used	used	used	used
10100	14	M=loadpipe(Ma,Mb)	----	----	(4x8)	----	----	----	used	used	used
10101	15	M=read_ram()	----	used	used	----	----	(ram)	used	used	----
10110	16	M=write_ram()	used	----	used	----	----	----	used	used	----
10111	17	M=linearstep(*)	used	used	used	----	----	(acc)	used	used	----
11000	18	M=macs(Ma,Mb)	----	----	(bus)	used	used	(acc)	----	----	----
11001	19	M=macb(Ma,Mb)	used	used	used	used	used	used	used	used	----
11100	1C	M=vectprod(Ma,Mb)	used	used	(2x16)	used	(sign)	used	used	used	----
11101	1D	M=complex(Ma,Mb)	used	used	(2x16)	used	(sign)	used	used	used	----
11110	1E	M=nop	----	----	----	----	----	----	----	----	----
11111	1F	M=halt	----	----	----	----	----	----	----	----	----

(*) = optional Mb operand for Range Unit: () or (Mb)

[0067] In general, with the data processing circuit 1, elementary functions are given directly with the instruction on an instruction-by-instruction basis: each cycle a different instruction is possible. These instructions do not use any information from the control register and operate without any side effects. The control registers are supplied to be able to fully exploit the possibilities of the hardware.

[0068] The multiplier and accumulator (MAC) control registers (cr12, cr13 and cr16) control the operations of the multiplier 23 and the accumulator 22. They control all the options of the accumulator 22 and range clipper in high performance block mode operations. Whether a value from a control register is used depends on each particular extended instruction. The table immediately above defines which part of the control registers is used by each function.

[0069] Other functional units of the data processing circuit 1 also include control registers, which can be accessed in a similar manner to extend the basic instructions.

~~[0044]~~ **[0070]** A product specification entitled "IMAGINE: The Image Engine -- Documentation & User's Manual" (version 2.80), provides additional details of embodiments of the data processing circuit 1 and is incorporated herein by reference and is appended as an annex to this specification.

ABSTRACT OF THE DISCLOSURE

The present invention provides a circuit for processing integer data, especially for graphic applications having a multiplier unit which includes a pipeline in which the word length is adjustable for multiplying integer data s words of 8 bits or multiples thereof an arithmetic logic unit (ALU) for performing arithmetic operations on integer data words, the word length of which is adjustable in 8 bits or multiples thereof; a register unit provided with at least two registers for storage of integer data words having multiples of 8 bits on which the operation and/or pipeline multiplication has to be performed; and a bus structure having a number of separate buses which effects the transport of integer data words from and to the multiplier unit, the arithmetic logic unit and the register unit.